AliSim: A Fast and Versatile Phylogenetic Sequence Simulator for the Genomic Era

Nhan Ly-Trong,[1] Suha Naser-Khdour,[2] Robert Lanfear,[2] and Bui Quang Minh*,[1]

[1]School of Computing, College of Engineering and Computer Science, Australian National University, Canberra, ACT 2600, Australia

[2]Ecology and Evolution, Research School of Biology, College of Science, Australian National University, Canberra, ACT 2600, Australia

***Corresponding author:** E-mail: m.bui@anu.edu.au.

Associate editor: Keith Crandall

Abstract

Sequence simulators play an important role in phylogenetics. Simulated data has many applications, such as evaluating the performance of different methods, hypothesis testing with parametric bootstraps, and, more recently, generating data for training machine-learning applications. Many sequence simulation programmes exist, but the most feature-rich programmes tend to be rather slow, and the fastest programmes tend to be feature-poor. Here, we introduce AliSim, a new tool that can efficiently simulate biologically realistic alignments under a large range of complex evolutionary models. To achieve high performance across a wide range of simulation conditions, AliSim implements an adaptive approach that combines the commonly used rate matrix and probability matrix approaches. AliSim takes 1.4 h and 1.3 GB RAM to simulate alignments with one million sequences or sites, whereas popular software Seq-Gen, Dawg, and INDELible require 2–5 h and 50–500 GB of RAM. We provide AliSim as an extension of the IQ-TREE software version 2.2, freely available at www.iqtree.org, and a comprehensive user tutorial at http://www.iqtree.org/doc/AliSim.

Key words: **sequence simulation, phylogenetics, molecular evolution.**

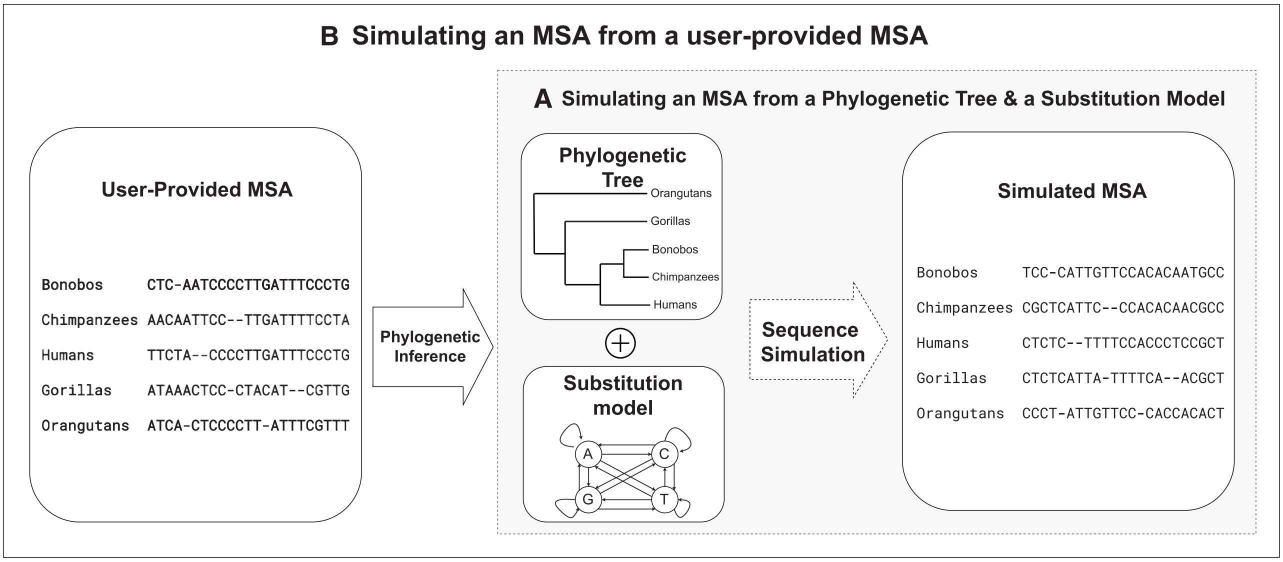
Introduction

Simulating multiple sequence alignments (MSAs) plays a vital role in phylogenetics. Sequence simulation has many applications, such as evaluating the performance of phylogenetic methods (Garland et al. 1993; Kuhner and Felsenstein 1994; Tateno et al. 1994; Huelsenbeck 1995), conducting parametric bootstraps, testing hypothesis (Goldman 1993a, 1993b; Adell and Dopazo 1994; Schoeniger and von Haeseler 1999), facilitating approximate Bayesian computation (Beaumont et al. 2002), and generating data for training machine-learning applications (Abadi et al. 2020; Leuchtenberger et al. 2020; Ling et al. 2020; Suvorov et al. 2020). Typical sequence simulation programmes (such as Seq-Gen (Rambaut and Grassly 1997), Dawg (Cartwright 2005), and INDELible (Fletcher and Yang 2009)) require the user to specify as input a tree and a model of sequence evolution to generate an alignment of sequences at the tips of the tree (fig. 1A).

Existing simulators often require long runtimes and a lot of memory to generate MSAs with millions of sequences or sites. The only exception to this is the recently-introduced phastSim (De Maio et al. 2022), designed to simulate alignments of hundreds of thousands of genomes from viruses such as SARS-CoV-2.

New Approaches

Here, we develop a fast, efficient, versatile, and realistic sequence alignment simulator called AliSim. Our simulator integrates a wide range of evolutionary models, available in the IQ-TREE software (Nguyen et al. 2015; Minh et al. 2020), including standard, mixture, partition, and insertion–deletion models (indels). In addition, AliSim can simulate MSAs that mimic the evolutionary processes underlying empirical alignments, a feature not available in other tools. AliSim allows the user to provide an input MSA, then infers the evolutionary process from that MSA, and subsequently simulates new MSAs from the inferred tree and model (fig. 1B). To further simplify this process, we also include the ability to simulate alignments based on the empirically-derived stationary distribution of nucleotides extracted from a large database (Naser-Khdour et al. 2021). To reduce the runtime across a wide range of simulation conditions, we implement a new adaptive approach that allows AliSim to dynamically switch between the rate matrix approach (also known as the Gillespie algorithm; Schoeniger and von Haeseler 1995; Fletcher and Yang 2009) and the probability matrix approach (also known as the matrix exponentiation method; Schoeniger and von Haeseler 1995) during the simulation. AliSim can simulate large alignments with millions



```
Bonobos     CTC-AATCCCCTTGATTTCCCTG
Chimpanzees AACAATTCC--TTGATTTTCCTA
Humans      TTCTA--CCCCTTGATTTCCCTG
Gorillas    ATAAACTCC-CTACAT--CGTTG
Orangutans  ATCA-CTCCCCTT-ATTTCGTTT
```

```
Bonobos     TCC-CATTGTTCCACACAATGCC
Chimpanzees CGCTCATTC--CCACACAACGCC
Humans      CTCTC--TTTTCCACCCTCCGCT
Gorillas    CTCTCATTA-TTTTCA--ACGCT
Orangutans  CCCT-ATTGTTCC-CACCACACT
```

Fig. 1. Sequence simulation process with two scenarios: (*A*) Simulating an MSA from a phylogenetic tree and a Markov substitution model, and (*B*) Simulating an MSA that mimics the underlying evolutionary process of a user-provided MSA. Here, the phylogenetic tree and the substitution model parameters are internally inferred from the user-provided MSA, which are used to simulate a new MSA.

of sequences and sites using much lower computing times and memory than existing tools. For example, AliSim consumes 1.3 GB RAM and 1.4 h to produce an MSA containing 1 million sequences with 30,000 sites per sequence, whereas INDELible, Seq-Gen, and Dawg require 2–5 h and 50–500 GB of RAM.

Results

AliSim Supports a Wide Range of Evolutionary Models

Table 1 compares the features of AliSim to other software. Notably, AliSim supports many evolutionary models not available in other software (table 1). AliSim allows users to simulate different data types, including DNA, amino acid, codon, binary, and multi-state morphological data using more than 200 time-reversible substitution models and 100 non-reversible models (Minh et al. 2020). AliSim also supports insertion–deletion models, as well as complex partition and mixture models. Moreover, users can specify model parameters or define new models via a short command-line option or a NEXUS file.

To model rate heterogeneity across sites, AliSim offers invariant sites, discrete and continuous Gamma distributions (Yang 1994; Gu et al. 1995), distribution-free rate models (Yang 1995; Soubrier et al. 2012), and the GHOST model (Crotty et al. 2020). AliSim also implements branch-specific substitution models, which assign different models of sequence evolution to individual branches of a tree. To mimic more complex evolutionary patterns, such as incomplete lineage sorting or recombination, AliSim extends the partition model by allowing different tree topologies for each partition.

AliSim Offers More Realistic Simulations

Scenario 1: Simulating MSAs that Mimic a User-Provided MSA

A common use-case for alignment simulation software is that users want to simulate an MSA that mimics the evolutionary history of a given MSA, for example, because this is needed for parametric bootstrap analysis. Until now, this required at least a two-step process whereby users first inferred the tree and model in one piece of software, then used these as input to the MSA simulation tool. The resulting MSA often failed to capture many characteristics of the original MSA, such as the position of gaps and the site-specific evolutionary rates. AliSim improves this process by first running IQ-TREE to infer an evolutionary model and a tree from the input MSA and then immediately generating any number of simulated MSAs from the inferred tree and model with the same gap patterns of the original MSA. For simulations under a mixture model, AliSim randomly assigns a model component of the mixture to each site according to the site posterior probability distribution of the mixture. For site-frequency mixture models, AliSim applies the posterior mean site frequencies (Wang et al. 2018). Similarly, AliSim employs the posterior mean site rates to better reflect the underlying evolutionary rate variation across sites. All these mechanisms help produce simulated MSAs that better reflect the relevant features of the original MSAs (*fig. 1B*).

Scenario 2: Simulating MSAs from a Random Tree and/or Random Parameters from Empirical/User-Defined Distributions

When using Seq-Gen or Dawg, users need to provide as input a tree with branch lengths. To avoid this sometimes cumbersome step, AliSim allows users to generate a

Table 1. Feature comparison between AliSim v2.2.0 (March 8, 2022) and existing tools, Seq-Gen v1.3.4 (August 29, 2019), Dawg v2.0.1 (March 8, 2022), INDELible v1.03, and phastSim v0.0.4 (February 8, 2022).

Features	Seq-Gen	Dawg	INDELible	phastSim	AliSim
Substitution models					
DNA	✓	✓	✓	✓	✓
Amino acid	✓	✓	✓		✓
Codon		✓	✓	✓	✓
Binary and discrete morphological					✓
RNA (base-pairing)		✓			
Non-reversible DNA and amino acid			✓	✓	✓
Models of rate heterogeneity across sites					
Invariant sites (+I)	✓	✓	✓	✓	✓
Discrete Gamma distribution (+G*k*)	✓		✓		✓
Continuous Gamma distribution (+GC)	✓	✓	✓	✓	✓
Distribution-free (+R*k*) (user-defined)				✓	✓
Codon-position-specific rates	✓				
Nonsynonymous/synonymous codon rate heterogeneity			✓	✓	
Complex models					
Insertion–deletion		✓	✓	✓	✓
Indel-rate variation				✓	
Partition	Same model*	✓	✓		✓
Site mixture**			Codon only		✓
Tree mixture for non-tree-like evolution	Same model and taxa	✓	✓		✓
Branch-specific substitutions***		✓	✓		✓
Hypermutability				✓	
Heterotachy (Crotty et al. 2020)					✓
Functional divergence (Gaston et al. 2011)					✓
User-defined models	✓	✓	✓	✓	✓
Ascertainment bias correction					✓
Biologically realistic simulations					
Mimicking a user-provided MSA					✓
Model parameters following empirical or user-defined distributions					✓
Simulating random trees			✓		✓
Other features					
Multifurcating trees		✓	✓	✓	✓
Branch length scaling	✓	✓	✓	✓	✓
Graphical user interface	✓				
Outputting ancestral sequences	✓	N/A	✓	✓	✓
Output format	PHYLIP, NEXUS, FASTA	PHYLIP, FASTA, NEXUS, CLUSTAL, POO	PHYLIP, FASTA, NEXUS	PHYLIP, FASTA, NEWICK, MAT, Info	PHYLIP, FASTA
Inserting output header	✓		✓		
Output compression					Gzip
Programming language	C	C++	C++	Python	C++

*, all partitions must share the same evolutionary model; **, a mixture model is a set of substitution models where each site has a probability of belonging to a substitution model; ***, users can specify different evolutionary models to individual branches of a tree.

random tree under biologically plausible models such as the Birth-Death model (Kendall 1948) and the Yule-Harding model (Yule 1925; Harding 1971). For the Yule-Harding model, users only need to specify the number of leaves of the tree. For the birth-death model, users need to additionally provide the speciation and extinction rate. Branch lengths are randomly generated from an exponential distribution with a user-adjustable default mean of 0.1 or from a user-defined distribution specified by a list of numbers.

Where users wish to simulate alignments that mimic empirical MSAs in the absence of a set of input alignments, AliSim can generate a stationary distribution of nucleotides from empirical distributions that were previously estimated from a large collection of empirical datasets (Naser-Khdour et al. 2021). Other parameters, such as substitution rates, non-synonymous/synonymous rate ratios, transition, and transversion rates, can be drawn from user-defined lists of numbers, allowing AliSim to incorporate arbitrary distributions for all simulation parameters.

AliSim Automatically Chooses a Simulation Method to Minimize the Runtime

Existing simulators typically employ either the rate matrix approach or the probability matrix approach to evolve sequences along a tree (see Methods). However, their performance varies with different sequence lengths (L) and branch lengths (t). Therefore, AliSim automatically switches between the rate matrix and probability matrix approaches to minimize the computing time. To determine when to use each approach, we compared the runtime of the rate matrix approach with the probability matrix approach on simulations using different combinations of *L* and *t* (see Methods).

The simulation results showed that the rate matrix approach is generally faster than the probability matrix approach when $L^*t < 2.226$ and $L^*t < 17.307$ for the discrete and continuous rate heterogeneity models, respectively. Therefore, the adaptive approach applies the rate matrix approach for those branches that satisfy this inequality; otherwise, it will apply the probability matrix approach.

AliSim is Fast and Efficient Across a Range of Conditions without Indels

We benchmarked AliSim against Seq-Gen, INDELible, Dawg, and phastSim for a range of common phylogenomic simulation conditions with and without indels. Specifically, we simulated "deep" data with 30K sites and 10K to 1M sequences, and we simulated "long" data with 30K sequences and 10K to 1M sites (see Methods). We note before presenting these results that phastSim is designed specifically to simulate data along trees with a large proportion of extremely short branches. The trees in these simulations do not match these conditions, and so one might expect phastSim to perform poorly here. We include phastSim here for completeness and present a comparison of phastSim and AliSim under the conditions for which phastSim was designed later.

Figure 2 shows the benchmarking results. In the simulations without indels, when the data sets were small, runtimes and memory usage were similar across all pieces of software. However, AliSim shows increasing advantages in runtimes and memory usage as the data sets get bigger. For example, for the deepest data set (30K sites and 1M sequences; fig. 2*A*), Seq-Gen, Dawg, INDELible, phastSim, and AliSim required 2, 3.2, 4.9, >24, and 1.4 h of runtime, respectively. And for the longest data set (30K sequences and 1M sites; fig. 2*C*), Seq-Gen, Dawg, INDELible, phastSim, and AliSim required 2.2, 2.1, 3.7, >24, and 1.4 h respectively. Thus, AliSim is a fast sequence simulator under a range of conditions.

AliSim shows dramatic improvements over other software in peak memory usage. Figures 2*B* and *D* show that these improvements become large even for fairly modestly-sized datasets. For the deepest dataset (30K sites and 1M sequences; fig. 2*B*), Seq-Gen, Dawg, INDELible, and AliSim required 56, 488, 502, and 1.3 GB RAM, respectively (phastSim peak memory usage was not recorded as it took >24 h to run). For the longest dataset (30K sequences and 1M sites; fig. 2*D*), Seq-Gen, Dawg, INDELible, and AliSim consumed 56, 534, 499, and 0.2 GB RAM, respectively (as above, phastSim was excluded because it took >24 h to run). Importantly, the memory usage of AliSim only grows sub-linearly with respect to the data set size. For the deep-data simulations, when increasing the number of sequences from 10K to 1M (100-fold), the RAM consumption of AliSim only increased from 137 MB to 1.3 GB (~10-fold increase, fig. 2*B*). For the long-data simulations, increasing the sequence length from 10K to 1M sites (100-fold) only increased the RAM usage marginally from 156 MB to 222 MB (less than a 2-fold increase, fig. 2*D*). This result is due to the memory saving techniques employed in AliSim (see Methods), which work particularly well in these simulations because they have relatively balanced tree shapes.

We also tested the performance of existing tools on simulating MSAs from SARS-CoV-2-like trees. These differ from the previous simulations because they contain a large proportion of extremely short branch lengths, a situation for which phastSim was explicitly designed. In SARS-CoV-2-like data simulations without indels (supplementary fig. S1*A*, Supplementary Material online), phastSim and AliSim were the two fastest pieces of software, requiring 10 and 14 min respectively, whereas Seq-Gen, Dawg, and INDELible took 1.8, 2.7, and 3.3 h, respectively, to simulate 1M sequences of 30K sites. In terms of RAM consumption, phastSim and AliSim only needed 1.4 and 1.3 GB RAM, respectively, whereas Seq-Gen, Dawg, and INDELible required 56, 482, and 502 GB RAM (supplementary fig. S1*B*, Supplementary Material online). We note that the performance of phastSim in these conditions is particularly remarkable because it is written in Python. Because of this, it may be that the language itself rather than the sequence simulation algorithms of phastSim limit its performance, and we speculate that phastSim may be able to be even faster if re-written in C or C++.

AliSim is Memory-Efficient in Simulations with Indels or Other Complex Models

We further compared the performance of AliSim, Dawg, and INDELible in simulations with insertion–deletion models. Seq-Gen and phastSim were excluded in this comparison because Seq-Gen does not support indels and phastSim only produces unaligned sequences. Due to excessive computation times by all software, we reduced the number of sequences by 100-fold. The results (fig. 2*E*, *F*, and supplementary fig. S1*C* and *D*, Supplementary Material online) showed that INDELible consumed a huge amount of memory and could only simulate up to 1K sequences for trees with normal branch lengths. Both Dawg and AliSim can complete all simulations. Dawg was 2.4–6.4 times faster than AliSim but consumed up to 95 times more memory.

Finally, we also tested other scenarios such as SARS-CoV-2-like data simulations with indels (supplementary fig. S1*E* and *F*, Supplementary Material online), discrete Gamma rate heterogeneity (supplementary fig. S2, Supplementary Material online) and codon models (supplementary fig. S3, Supplementary Material online). In terms of runtimes, AliSim is up to 1.7 times slower than the fastest software (Dawg) in simulating codon data with indels; but up to 7 times faster than the second-fastest software (Seq-Gen or Dawg) in all other simulations. In terms of

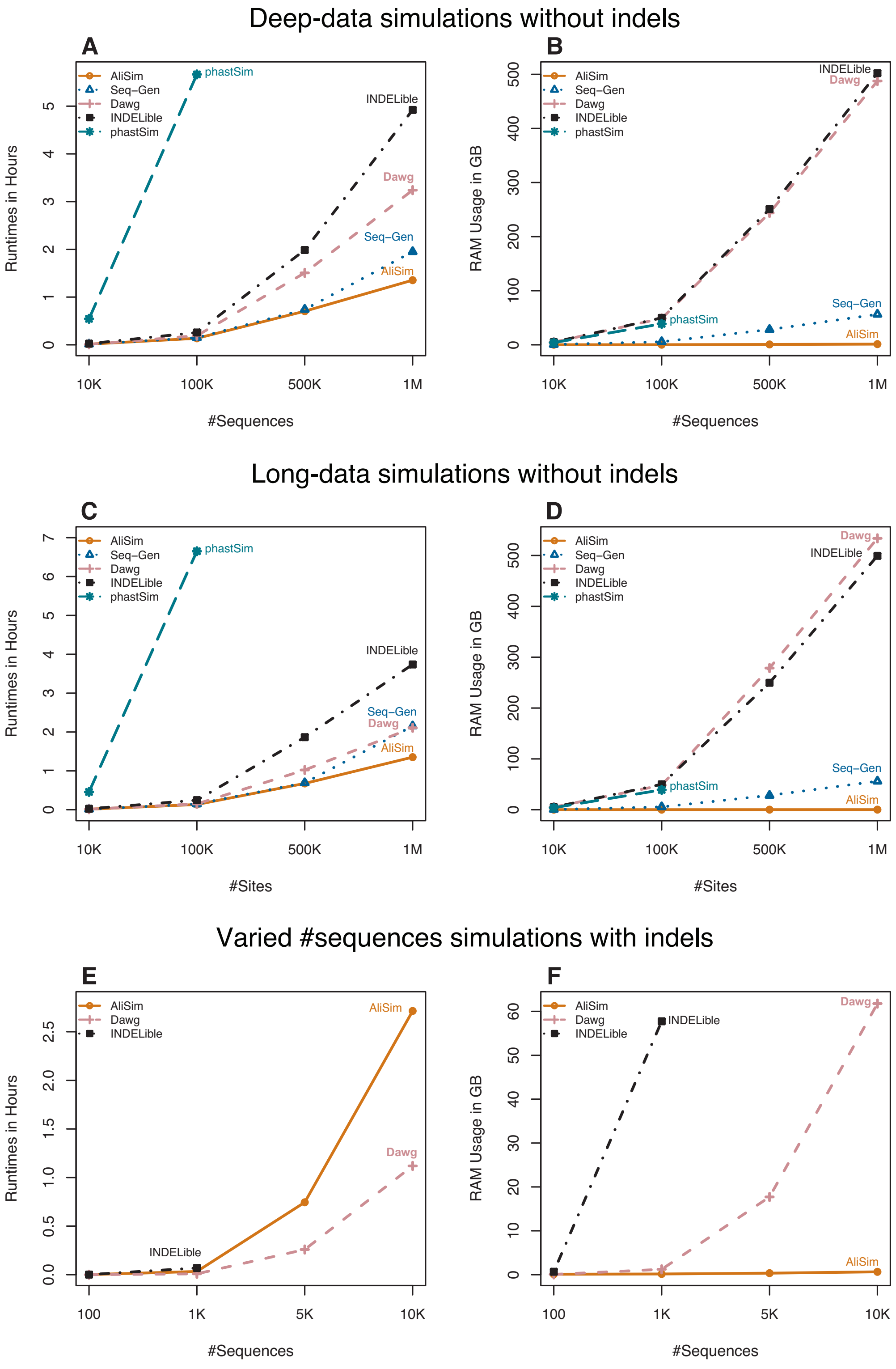


Fig. 2. Runtimes and peak memory consumptions of five software AliSim, Seq-Gen, Dawg, INDELible, and phastSim for deep-data (varying number of sequences and 30K sites; sub-panels *A* and *B*) simulations without indels, long-data (varying number of sites and 30K sequences; sub-panels *C* and *D*) simulations without indels, and varied #sequences (varying number of sequences and setting root sequence length at 30K sites; sub-panels *E* and *F*) simulations with indels.

memory consumption, AliSim was always the most efficient in all settings, using up to 880 times less RAM than the second-best piece of software.

The Efficiency of the Adaptive Algorithm

The adaptive approach helps AliSim achieve high performance by selecting the most efficient simulation approach for each branch. For example, in simulations under trees where branch lengths were generated from an exponential distribution with a mean of 0.1, the adaptive method applies the probability matrix approach rather than the rate matrix approach on most branches, simply because most branches are longer than the switching parameters. The benefits of the adaptive approach can be measured in our simulations by forcing AliSim to use one method. For example, using the adaptive approach, AliSim took only 1.4 h to simulate 1M sequences of 30K sites (fig. 2A). However, if we force AliSim to employ only the rate matrix approach, it takes more than 5 h to simulate the same data set. Similarly, the adaptive approach took only 14 min to simulate a data set on a SARS-CoV-2-like tree (supplementary fig. S1A, Supplementary Material online), but if we force AliSim to use the probability matrix approach, the same simulation takes 1.4 h.

Software Validation

To validate the AliSim, we simulated 287 MSAs with 100 sequences across a wide range of substitution models and insertion–deletion rates of 0.0, 0.02, 0.04, 0.06, 0.08, and 0.1. These choices of indel rates follow empirical studies (Cartwright 2009). We then ran IQ-TREE to determine the best-fit model using ModelFinder (Kalyaanamoorthy et al. 2017) and reconstructed phylogenetic trees under the best-fit model. We compared the topology between the true trees and the inferred trees using the Robinson-Foulds distance (Robinson and Foulds 1981).

Supplementary table S1, Supplementary Material online shows that in 148 tests (51.57%), the true model was recovered as the best-fit model. In 243 tests (84.67%), 246 tests (85.71%), and 267 tests (93.03%), the true models appear in the top-2, top-3, and top-4 best models, respectively. The average Robinson-Foulds distance between the true trees and the inferred trees across all test cases was 2.06 (s.e. 0.135). That means the inferred trees differed from the true trees by only 1.03 of 97 (1.06%) internal branches. The tree lengths (sum of branch lengths) of the inferred trees differed from the true trees by only 1.9%.

For simulations with non-zero insertion–deletion rates, the average differences in the alignment length and proportion of gaps between MSAs simulated by AliSim and those by INDELible were 0.52% and 0.25%, respectively (supplementary table S2, Supplementary Material online).

Conclusion

In conclusion, AliSim is a fast and memory-efficient simulation tool, which simplifies and speeds up many common workflows in phylogenetics. AliSim offers a very broad spectrum of simulation features. Thanks to a small memory footprint, AliSim can simulate even very large alignments on personal computers.

Materials and Methods

We developed AliSim in C++ as an extension to the IQ-TREE software to take advantage of all models of sequence evolution provided in IQ-TREE. Generally, AliSim works by first generating a sequence at the root of the tree following the stationarity of the model. AliSim then recursively traverses along the tree to generate sequences at each node of the tree based on the sequence of its ancestral node. AliSim completes this process once all the sequences at the tips are generated. In the following, we introduce general notations and three simulation approaches to simulate sequence evolution along a branch of a tree in a general case with indels.

Let $Q = (q_{xy})$ be a rate matrix of a Markov model, where $x, y \in \Sigma$, a finite alphabet, for example, the alphabet of nucleotides or amino acids; q_{xy} is the instantaneous substitution rate from x to y. Q is normalized such that the row sum is zero: $q_{xx} = -\sum_{y \neq x} q_{xy}$ and the total substitution rate is one: $\sum_{x} \pi_x q_{xx} = -1$, where π_x is the state frequency. For stationary models, we normalize Q by the equilibrium state frequencies, but for non-stationary models such as branch-specific models, Q is normalized by the state frequencies of the corresponding branch. Additionally, we assume a model of rate heterogeneity across sites, such as the invariant site proportion, the continuous/discrete Gamma model (Yang 1994), or the distribution-free rate model (Yang 1995; Soubrier et al. 2012). Let r_I, r_D be the insertion and deletion rate, respectively, relative to the substitution rate. Let Φ_I, Φ_D be the insertion-length and deletion-length distributions, respectively. AliSim allows users to use built-in indel-length distributions, such as Geometric, Negative Binomial, Zipfian, and Lavalette distributions (Fletcher and Yang 2009), or specify their own distributions. By default, AliSim uses a Zipfian distribution with an exponent of 1.7 as previously estimated from empirical data (Benner et al. 1993; Cartwright 2009). Given a sequence $X = x_1 x_2 \ldots x_L,\ x_i \in \Sigma \cup \{-\}$ (where $'-'$ denotes the gap character), at an ancestral node of a phylogenetic tree; a vector of site-specific rate $R = r_1 r_2 \ldots r_L$, r_i is generated according to the site-rate heterogeneity model; and a branch length t, as the number of substitutions per site, of the branch connecting the ancestral node to a descendent node, we now describe three approaches to generate a new sequence $Y = y_1 y_2 \ldots y_{L'},\ y_i \in \Sigma \cup \{-\}$, at the descendent node. L' might be different from L if the insertion rate is non-zero.

The rate matrix approach: This approach implements the Gillespie algorithm (Gillespie 1977) as follows. We compute the total mutation rate for the ancestral sequence X as the sum of site-specific mutation rates: $M = S + I + D$, where S, I, D is the total rate of substitutions, insertions, and deletions of all sites respectively. $S = -\sum_{i=1}^{L} q_{x_i x_i} r_i;\ \ I = r_I(L+1);\ \ D = r_D(L - 1 + u_D)$, where u_D is the mean of the deletion size distribution (Cartwright 2005).

0) Set $Y \leftarrow X$ and $L' \leftarrow L$.
1) Generating a waiting time w for a mutation to occur from an exponential distribution with a mean of $\frac{1}{M}$.
2) If $w > t$, no mutation occurs, then we stop and return Y as the sequence at the descendent node.
3) If $w \leq t$, a mutation occurs and we randomly select a mutation type as substitution, insertion, or deletion with probabilities of $\frac{S}{M}$, $\frac{I}{M}$, $\frac{D}{M}$ respectively.
4) If the mutation type is substitution:

- 4.1. Randomly select a non-gap position i, $1 \leq i \leq L^{'}$ where the substitution occurs with probabilities $\frac{-q_{y_iy_i}\, r_i}{S}$. If Y contains only gaps, we terminate the algorithm.
- 4.2. Randomly choose a new state z_i according to probabilities $\frac{q_{y_iz_i}}{-q_{y_iy_i}}$.
- 4.3. Update the total substitution rate to reflect the new state: $S \leftarrow S + (q_{y_iy_i} - q_{z_iz_i})r_i$.
- 4.4. Assign $y_i \leftarrow z_i$ and go to step 7.

5) If the mutation type is insertion:
   - 5.1. Uniformly select a non-gap position i, $1 \leq i \leq L^{'} + 1$ where the insertion occurs.
   - 5.2. Randomly generate a new sequence $Z = z_1 \ldots z_j$ based on the stationary distribution of the model, where the sequence length j follows the insertion-length distribution Φ_I.
   - 5.3. Insert Z into Y at position i. If $i = L^{'} + 1$, Z is appended at the end of Y.
   - 5.4. Insert a stretch of j gaps into position i of the sequences at all other nodes of the tree so that all sequences have the same length.
   - 5.5. Generate a vector of site rates $(s_1, \ldots, s_j)$ according to the distribution of rate heterogeneity across sites and insert this vector into R, at position i.
   - 5.6. Update the sequence length $L^{'} \leftarrow L^{'} + j$, the total substitution rate $S \leftarrow S - \sum_{i=1}^{j} q_{z_iz_i} s_i$, the total insertion rate $I \leftarrow I + r_I j$, and the total deletion rate $D \leftarrow D + r_D j$.
   - 5.7. Go to step 7.

6) If the mutation type is deletion:
   - 6.1. Generate a deletion length, j, from the deletion-length distribution Φ_D.
   - 6.2. Uniformly select a non-gap position i, $1 \leq i \leq L^{'} - j + 1$, where the deletion occurs.
   - 6.3. Initialize $P = \{p_1, p_2, \ldots, p_j\}$, a set of j non-gap positions in Y starting at position i. Note that p_j might be greater than $i + j$ if there are gaps between position i and $i + j$.
   - 6.4. Update the total substitution rate $S \leftarrow S + \sum_{i \in P} q_{y_iy_i} r_i$; then $\forall i \in P$, we set $y_i \leftarrow {}^{'} - {}^{'}$ and $r_i \leftarrow 0$.
   - 6.5. Update the total insertions rate $I \leftarrow I - r_I j$, and the deletion rate $D \leftarrow D - r_D j$.

7) Update the total mutation rate: $M \leftarrow S + I + D$ and the time $t \leftarrow t - w$. Go back to step 1.

The probability matrix approach: Instead of generating a series of waiting times, the probability matrix approach generates a new state y_i for each site in the sequence based on the state x_i, $i = 1, 2, \ldots, L$. For each site i, we compute the transition probability matrix $P(t, r_i) = e^{Qtr_i}$. Then, the new state y_i is drawn from the probability distribution $P_{x_iy_i}(t, r_i)$, $y_i \in \Sigma$. Note that when using a discrete rate model with k categories, we only need to compute $P(t, r_i)$ exactly k times to save computations. Whereas for a continuous Gamma rate model, we have to compute $P(t, r_i)$ for each site independently. After processing substitutions with the probability matrix approach, to simulate indels, we apply the Gillespie algorithm as described above on the new sequence Y without considering substitutions by setting and maintaining the total substitution rate S at zero.

The adaptive approach: In simulations without indels, the probability matrix approach has a time complexity independent of branch lengths, but the time complexity for the rate matrix approach grows with increasing branch lengths. In simulations with indels, the branch lengths affect the runtime of the rate matrix approach more significantly than that of the probability matrix approach. We expect the rate matrix approach to outperform the probability matrix approach for small t but the opposite for large t. Therefore, we derived an adaptive approach, in which we determined a switching parameter from the sequence length. For all branches where the branch length is smaller than this parameter, we employ the rate matrix approach. For the remaining (long) branches, we use the probability matrix approach. That means our adaptive algorithm will automatically switch between these two approaches on a per-branch basis to minimize the computations.

To determine the switching parameter, we performed simulations with different sequence lengths, ranging from 1K to 100K sites (a total of 19 tests), with/without rate heterogeneity. We measured the runtimes of the probability matrix approach when simulating MSAs under a random Yule-Harding tree with 10K tips based on the general time-reversible (GTR) model (Tavaré and Miura 1986) with/without continuous Gamma rate heterogeneity using a Gamma shape of 0.5. For each test case, we applied binary search on a predefined range of branch length to determine the switching parameter where the runtime of the rate matrix approach is less than the probability matrix approach (supplementary table S3, Supplementary Material online). We then determine the switching parameters using a least square fit across the simulations.

Memory Optimization Techniques

Naively, when simulating sequences along a bifurcating tree with n tips, we need to store up to ($2n$–1) sequences, consisting of $(n - 1)$ internal nodes and n tips. To save memory, we release the memory allocated to the sequence of an internal node if the sequences of its children nodes are already generated. In addition, in simulations without partitions, AliSim writes out the tip sequences to the output file immediately after simulating them, then frees the memory. This approach considerably reduces the maximum number of sequences that need to be stored in memory from $(2n - 1)$ to the maximum depth of the tree. For a balanced bifurcating tree, this maximal depth is $log_2(n) + 1$, leading to a substantial reduction in memory usage. But in the worst case of a completely unbalanced tree, the tree depth is *n* and we still save half of the memory. Hence, the memory saving depends on the tree shape.

Benchmark Experimental Set-up

We benchmarked AliSim against Seq-Gen, Dawg, INDELible, and phastSim. The benchmark was run on a Linux server with 2.0 GHz AMD EPYC 7501 32-Core Processor and 1-TB RAM. All simulators generated alignments in PHYLIP format. We ran all software in single threaded mode. Inspired by the newly emerged SARS-CoV-2 data, we tested the ability of all tools to simulate alignments with 30K sites and 10K, 100K, 500K, and 1M DNA sequences. For the model of sequence evolution, we applied the GTR model with a 0.2 proportion of invariant sites and a continuous Gamma model of rate heterogeneity across sites (shape parameter of 0.5). We ran different software to simulate sequences along random trees, drawn under the Yule-Harding model and exponentially distributed branch lengths with a mean of 0.1. We call this the deep-data simulation. Moreover, to mimic the size of real phylogenomic datasets, we simulated MSAs with 30K sequences and increased sequence length from 10K to 1M sites. This is called long-data simulation. For simulations with indels, we applied empirical parameters (Graur et al. 1989; Gu and Li 1995; Cartwright 2009) with the insertion and deletion rates of 0.03 and 0.09, respectively, and the indel-lengths drawn from a truncated Zipfian (Power-Law) distribution (Fletcher and Yang 2009) ($a = 1.7$; max = 50). Note that for INDELible, we used Method 2, which is more efficient than Method 1 in simulations with continuous rate heterogeneity across sites (Fletcher and Yang 2009). For phastSim, we used the "hierarchical" approach because the "vanilla" algorithm does not support continuous rate variation.

Supplementary Material

Supplementary data are available at *Molecular Biology and Evolution* online.

Acknowledgments

This work was supported by a Chan-Zuckerberg Initiative grant for open source software for science to B.Q.M. and R.L., an Australian Research Council Discovery Grant (DP200103151 to R.L. and B.Q.M.), a Moore-Simons Foundation grant 735923LPI (https://doi.org/10.46714/735923LPI) to B.Q.M., and partly by a Vingroup Science and Technology Scholarship (VGRS20042M to N.L.T.). The computational results have been obtained on the cluster at the Center for Integrative Bioinformatics Vienna (CIBIV). We thank Arndt von Haeseler for valuable comments and providing access to the CIBIV cluster; Caitlin Cherryh, Yu Lin, Fred Jaya, Weiwen Wang for their comments on the manuscript; Andrew Roger and Edward Susko for helpful discussions.

Data Availability

The data underlying this article are available in the Supplementary Material Online and the Zenodo Repository, at https://doi.org/10.5281/zenodo.6361862.

References

Abadi S, Avram O, Rosset S, Pupko T, Mayrose I. 2020. ModelTeller: model selection for optimal phylogenetic reconstruction using machine learning. *Mol Biol Evol.* **37**(11):3338–3352.

Adell JC, Dopazo J. 1994. Monte Carlo simulation in phylogenies: an application to test the constancy of evolutionary rates. *J Mol Evol.* **38**(3):305–309.

Beaumont MA, Zhang W, Balding DJ. 2002. Approximate Bayesian computation in population genetics. *Genetics* **162**(4): 2025–2035.

Benner SA, Cohen MA, Gonnet GH. 1993. Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *J Mol Biol.* **229**(4):1065–1082.

Cartwright RA. 2005. DNA assembly with gaps (Dawg): simulating sequence evolution. *Bioinformatics* **21**(Suppl. 3):31–38.

Cartwright RA. 2009. Problems and solutions for estimating indel rates and length distributions. *Mol Biol Evol.* **26**(2):473–480.

Crotty SM, Minh BQ, Bean NG, Holland BR, Tuke J, Jermiin LS, von Haeseler A. 2020. GHOST: recovering historical signal from heterotachously evolved sequence alignments. *Syst Biol.* **69**(2): 249–264.

De Maio N, Boulton W, Weilguny L, Walker CR, Turakhia Y, Corbett-Detig R, Goldman N. 2022. phastSim: efficient simulation of sequence evolution for pandemic-scale datasets. *PLoS Comput Biol.* **18**(4):e1010056.

Fletcher W, Yang Z. 2009. INDELible: a flexible simulator of biological sequence evolution. *Mol Biol Evol.* **26**(8):1879–1888.

Garland T, Dickerman AW, Janis CM, Jones JA. 1993. Phylogenetic analysis of covariance by computer simulation. *Syst Biol.* **42**(3): 265–292.

Gaston D, Susko E, Roger AJ. 2011. A phylogenetic mixture model for the identification of functionally divergent protein residues. *Bioinformatics* **27**(19):2655–2663.

Gillespie DT. 1977. Exact stochastic simulation of coupled chemical reactions. *J Phys Chem.* **81**(25):2340–2361.

Goldman N. 1993a. Statistical tests of models of DNA substitution. *J Mol Evol.* **36**(2):182–198.

Goldman N. 1993b. Simple diagnostic statistical tests of models for DNA substitution. *J Mol Evol* **37**(6):650–661.

Graur D, Shuali Y, Li WH. 1989. Deletions in processed pseudogenes accumulate faster in rodents than in humans. *J Mol Evol.* **28**(4): 279–285.

Gu X, Fu Y-X, Li W-H. 1995. Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites. *Mol Biol Evol.* **2**(4):546–557.

Gu X, Li WH. 1995. The size distribution of insertions and deletions in human and rodent pseudogenes suggests the logarithmic gap penalty for sequence alignment. *J Mol Evol.* **40**(4):464–473.

Harding EF. 1971. The probabilities of rooted tree-shapes generated by random bifurcation. *Adv Appl Probab.* **3**(1):44–77.

Huelsenbeck JP. 1995. Performance of phylogenetic methods in simulation. *Syst Biol.* **44**(1):17–48.

Kalyaanamoorthy S, Minh BQ, Wong TKF, von Haeseler A, Jermiin LS. 2017. ModelFinder: fast model selection for accurate phylogenetic estimates. *Nat Methods* **14**(6):587–589.

Kendall DG. 1948. On the generalized "birth-and-death" process. *Ann Math Stat.* **19**(1):1–15.

Kuhner MK, Felsenstein J. 1994. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol Biol Evol.* **11**(3):459–468.

Leuchtenberger AF, Crotty SM, Drucks T, Schmidt HA, Burgstaller-Muehlbacher S, von Haeseler A. 2020. Distinguishing felsenstein zone from farris zone using neural networks. *Mol Biol Evol*. **37**(12):3632–3641.

Ling C, Cheng W, Haoyu Z, Zhu H, Hua Z. 2020. Deep neighbor information learning from evolution trees for phylogenetic likelihood estimates. *IEEE Access* **8**:220692–220702.

Minh BQ, Schmidt HA, Chernomor O, Schrempf D, Woodhams MD, von Haeseler A, Lanfear R. 2020. IQ-TREE 2: new models and efficient methods for phylogenetic inference in the genomic era. *Mol Biol Evol*. **37**(5):1530–1534.

Naser-Khdour S, Minh BQ, Robert L. 2021. The influence of model violation on phylogenetic inference: a simulation study. *bioRxiv*.

Nguyen LT, Schmidt HA, von Haeseler A, Minh BQ. 2015. IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Mol Biol Evol*. **32**(1):268–274.

Rambaut A, Grassly NC. 1997. Seq-gen: an application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. *Bioinformatics* **13**(3):235–238.

Robinson DF, Foulds LR. 1981. Comparison of phylogenetic trees. *Math Biosci*. **53**(1–2):131–147.

Schoeniger M, von Haeseler A. 1995. Simulating efficiently the evolution of DNA sequences. *Bioinformatics* **11**(1):111–115.

Schoeniger M, von Haeseler A. 1999. Toward assigning helical regions in alignments of ribosomal RNA and testing the appropriateness of evolutionary models. *J Mol Evol*. **49**:691–698.

Soubrier J, Steel M, Lee MSY, Der Sarkissian C, Guindon S, Ho SYW, Cooper A. 2012. The influence of rate heterogeneity among sites on the time dependence of molecular rates. *Mol Biol Evol*. **29**(11): 3345–3358.

Suvorov A, Hochuli J, Schrider DR. 2020. Accurate inference of tree topologies from multiple sequence alignments using deep learning. *Syst Biol*. **69**(2):221–233.

Tateno Y, Takezaki N, Nei M. 1994. Relative efficiencies of the maximum-likelihood, neighbor-joining, and maximum-parsimony methods when substitution rate varies with site. *Mol Biol Evol*. **11**(2):261–277.

Tavaré S, Miura RM. 1986. Some probabilistic and statistical problems in the analysis of DNA sequences. *Lect Math Life Sci*. **17**: 57–86.

Wang HC, Minh BQ, Susko E, Roger AJ. 2018. Modeling site heterogeneity with posterior mean site frequency profiles accelerates accurate phylogenomic estimation. *Syst Biol*. **67**(2): 216–235.

Yang Z. 1994. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *J Mol Evol*. **39**(3):306–314.

Yang Z. 1995. A space-time process model for the evolution of DNA sequences. *Genetics* **139**(2):993–1005.

Yule GU. 1925. A mathematical theory of evolution based on the conclusions of Dr. J. C. Willis, F.R.S. *Philos Trans R Soc Lond Ser B, Contain Pap a Biol Character*. **213**:21–87.